# Molecular Learning of wDNF Formulae

Byoung-Tak Zhang and Ha-Young Jang

Biointelligence Laboratory, Seoul National University, Seoul 151-742, Korea
{btzhang, hyjang}@bi.snu.ac.kr
http://bi.snu.ac.kr/

**Abstract.** We introduce a class of generalized DNF formulae called wDNF or weighted disjunctive normal form, and present a molecular algorithm that learns a wDNF formula from training examples. Realized in DNA molecules, the wDNF machines have a natural probabilistic semantics, allowing for their application beyond the pure Boolean logical structure of the standard DNF to real-life problems with uncertainty. The potential of the molecular wDNF machines is evaluated on real-life genomics data in simulation. Our empirical results suggest the possibility of building error-resilient molecular computers that are able to learn from data, potentially from wet DNA data.

## 1 Introduction

Disjunctive normal form (DNF) is a disjunction of conjunctions of Boolean variables, such as ($x_1$ AND $x_2$) OR ($x_1$ AND $\bar{x}_3$) OR ($x_2$ AND $x_3$) where $x_i$ represent attributes or binary-valued variables and $\bar{x}_i$ are their negations. The conjunctions in the form of ($x_1$ AND $x_2$) are called terms. DNF offers an interesting structure for representing knowledge in a logical form. For example, any Boolean function can be represented by a finite set of terms. Although previous research shows that the $k$-term DNF, i.e. DNF having $k$ terms at most, is learnable with attribute noise if the noise rate is known exactly [10, 8], the pure Boolean logical nature of DNF restricts its application [11].

Here we introduce a generalized form of DNF that is more resilient to noisy and/or incomplete data thus applicable beyond the pure logical problems. This weighted DNF or wDNF formula extends DNF twofold. On the conjunction level the attributes can appear multiple times, e.g. $x_1x_1 = x_1^2$ as well as $x_1$. This allows for higher-order attributes, enhancing the expressive power of DNF (We hurry to mention that logically $x_1$ AND $x_1 = x_1$, but this is true only when the variable does not contain noise). On the disjunction level the terms are permitted to appear multiple times. Thus the entire formula is a "disjunctive ensemble of conjunctions of higher-order terms". The number of copies of the terms represents the weight of voting in decision making, hence the "weighted DNF".

We show that the wDNF formulae can be learned from training examples using DNA computing, resulting in molecular wDNF machines. The probabilistic nature of the computation performed by the molecular wDNF machines is discussed along with its robustness against uncertainty arising from both internal (e.g., molecular reaction) and external (e.g., data) sources. The general setting for learning the wDNF machines is similar to the genetic programming (GP) framework where the programs for digital computers are evolved using the principle of natural selection [5, 14, 7]. While GP evolves tree-structured expressions, here we evolve the wDNF expressions encoded in DNA molecules. Starting from a random combinatorial library of wDNF expressions our "molecular programming" (MP) method evolves a wDNF formula that best fits to the learning sample. The potential of the molecular wDNF machines is evaluated on a DNA-based diagnosis problem. The terms in wDNF in this particular case represent the conjunctive rules of DNA markers for diagnosing a leukemia. Our simulation studies demonstrate a robust and highly competitive performance of the molecular wDNF machines on this real-life data.

The paper is organized as follows. Section 2 presents a formal description of the wDNF form. Section 3 presents the molecular algorithm for learning a wDNF formula. Section 4 explains the probabilistic nature of the molecular wDNF machines based on statistical mechanics of DNA hybridization reactions. Section 5 shows the simulation results on the diagnosis problem. We also evaluate the robustness of wDNF formulae by analyzing wDNF formulae containing small portions of the complete combinatorial terms. Section 6 draws conclusions.

## 2 Weighted Disjunctive Normal Form (wDNF)

Let $x_i$ denote an attribute or a Boolean variable, i.e. $x_i \in \{0, 1\}$. A literal consists of a variable $x_i$ or its negation $\bar{x}_i$. The former is called a positive literal and the latter a negative literal. For notational simplicity, the negative literal can be considered as a new positive literal by renaming it as $x_j = \bar{x}_i$. We shall adopt this convention in the following, unless otherwise noted. More generally, we consider the powers of literals and denote a literal of degree $r$ by $x_i^r$, where $r$ is an integer.

Then, a term is defined as a conjunction of the (positive) literals of degree one:

$$C_i = (x_{i_1}, x_{i_2}, \cdots, x_{i_k}, \cdots, x_{i_{n_i}}) = x_{i_1} x_{i_2} \cdots x_{i_k} \cdots x_{i_{n_i}}, \tag{1}$$

where $x_{i_k} \in \{x_1, x_2, ..., x_n\}$. For example, $C_i = (x_{i_1}, x_{i_2}, x_{i_3}) = x_1 x_4 x_5$ represents a term consisting of three literals $x_1$, $x_4$ and $x_5$. In general, the number of variables $n_i$ in a term $C_i$ may vary. A disjunctive normal form, DNF, on $n$ literals is defined as the disjunction of the terms:

$$DNF = \{C_1, C_2, \cdots, C_j, \cdots, C_N\} = C_1 + C_2 + \cdots + C_j + \cdots + C_N, \tag{2}$$

where $C_j$ is a term of an arbitrary number of literals out of $x_1, ..., x_n$. A $k$-term DNF formula is a DNF formula with maximum $k$ terms. For instance, $\{x_1 x_2 x_3, x_4 x_5, x_1 x_3 x_5\}$ is an example of a 3-term DNF formula on five literals $x_1, x_2, x_3, x_4, x_5$.

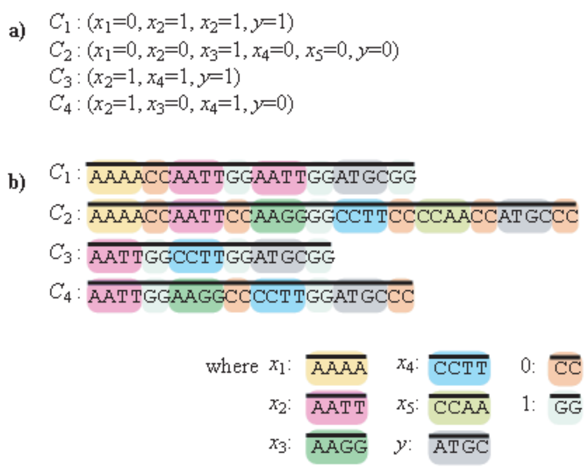
a)
C1 : (x1=0, x2=1, x2=1, y=1)
C2 : (x1=0, x2=0, x3=1, x4=0, x5=0, y=0)
C3 : (x2=1, x4=1, y=1)
C4 : (x2=1, x3=0, x4=1, y=0)
b)
C1 : AAAACCAATTGGAATTGGATGCGG
C2 : AAAACCAATTCCAAGGGGCCTTCCCCAACCATGCCC
C3 : AATTGGCCTTGGATGCGG
C4 : AATTGGAAGGCCCCTTGGATGCCC
where x1: AAAA
x2: AATT
x3: AAGG
x4: CCTT
x5: CCAA
y: ATGC
0: CC
1: GG

**Fig. 1.** A wDNF formula in two different representations: (a) a collection of terms, (b) a library of DNA molecules corresponding to (a). The DNA code shown is illustration-purposes only.

The weighted DNF (wDNF) generalizes the DNF in two ways. First, at the conjunction level the terms can be of higher degree $r$. Second, at the disjunction level a term can appear $w$ copies. Note that in Eqn. (1) all the literals in DNF are of maximum degree 1. In wDNF the term is generalized to contain literals of arbitrary degree. A term of degree $r$ is defined then a conjunction of literals of the form:

$$C_i = (x_{i_1}^{r_{i_1}}, x_{i_2}^{r_{i_2}}, \cdots, x_{i_k}^{r_{i_k}}, \cdots, x_{i_{n_i}}^{r_{i_{n_i}}}) = x_{i_1}^{r_{i_1}} x_{i_2}^{r_{i_2}} \cdots x_{i_k}^{r_{i_k}} \cdots x_{i_{n_i}}^{r_{i_{n_i}}} \tag{3}$$

where $x_{i_j} \in \{x_1, x_2, ..., x_n\}$ and $r_{i_j} \leq r$, $j = 1, ..., n_i$ for fixed $r$. For example, $C_i = (x_{i_1}^2, x_{i_2}^3, x_{i_3}^1) = x_{i_1}^2 x_{i_2}^3 x_{i_3}^1 = x_{i_1} x_{i_1} x_{i_2} x_{i_2} x_{i_2} x_{i_3}$ represents a term of degree 3. The generalized term is satisfied only if every occurrence of the literal is bound to a "sample" value. There are the designated instantiated "samples" of each literal.

Using the terms of degree $r$ on $n$ literals, a wDNF formula is defined as:

$$\begin{aligned} wDNF &= \{w_i C_1, w_2 C_2, \cdots, w_j C_j, \cdots, w_N C_N\} \\ &= w_1 C_1 + w_2 C_2 + \cdots + w_j C_j + \cdots + w_N C_N, \end{aligned} \tag{4}$$

where $w_j C_j$ means $w_j$ copies of the term $C_j$. The coefficient $w_j$ is interpreted to represent the "weight" or strength of the term. Thus, the number of variables matter in the generalized terms and the wDNF formulae.

To be more concrete, consider a wDNF formula for DNA-based diagnosis of disease shown in Figure 1(a). In this case the wDNF consists of four terms $C_1, ..., C_4$. A term is said to be "instantiated" if the values of the variables are bound to specific values. The instantiated term $C_1 = (x_1 = 0, x_2 = 1, x_2 =$

$1, y = 1)$, for example, encodes a diagnosis rule, where $y = 1$ indicates the label for disease. The meaning is that a DNA sample is decided positive ($y = 1$) if it contains two of the DNA marker 2 ($x_2 = 1$, $x_2 = 1$) and does not contain the DNA marker 1 (other variables do not care for this decision). This procedure can be implemented by hybridization reaction of complementary DNA molecules. For example, bead separation can be used to check whether the required values are contained or not. In the following, unless otherwise stated, we shall assume every term in wDNF has a label variable $y$ in it while other $x$-variables may appear or not. This does not lose the generality of the method since $y$-variable can be incorporated as an extra $x$-variable, but it makes the presentation more readable.

As shown in Figure 1 we encode the value of each variable as a DNA oligomer. For example, if we assume $x_1 = 0$ be encoded as a 6-mer like 'AAAACC', where 'AAAA' represents $x_1$ and 'CC' denotes the value 0. In this encoding scheme, a term consisting of 10 literals in total can be encoded as a 60-mer DNA.

## 3 Learning a wDNF Formula

In this section we describe the molecular algorithm for learning the wDNF formulae. The theoretical backgrounds of this procedure is given in the next section.

The goal is to learn a wDNF formula that best fits to a data set. We assume the training set $D$ of $K$ labeled DNA samples be given in the form

$$\begin{aligned} &D = \{(\mathbf{x}_i, y_i)\}_{i=1}^{K} \\ &\mathbf{x}_i = (x_{i_1}, x_{i_2}, ..., x_{i_n}) \in \{0,1\}^n \\ &y_i \in \{0,1\}, \end{aligned}$$

where $\mathbf{x}_i$ is the sample data and $y_i$ is the associated label. In the DNA-marker-based diagnosis problem, a training example $(10101, 1)$ means the sample is diagnosed positive ($y = 1$) if it contains the DNA markers numbered 1, 3, and 5 ($x_1 = 1, x_3 = 1, x_5 = 1$) and does not contain the rest ($x_2 = 0, x_4 = 0$). Figure 1 shows an example in DNA encoding.

To learn the formula we initialize a library of DNA molecules representing random combinatorial wDNF terms as shown in Figure 2. Given a query pattern $\mathbf{x}_q$ we extract from the library all the molecules (terms) that match the query. The extraction can be implemented using hybridization reaction in the same way to check which markers exist. The idea is to chop the query sequence into subsequences for individual variables. These chopped query sequences hybridize with the wDNF formula in the library. Only the fully double-stranded sequences are then separated (by selecting out the single-stranded sequences by beads).

These molecules will have class labels from which we decide the majority label as the class of the query pattern. To perform the matching between $\mathbf{x}_i$ and $\mathbf{x}_q$ for $i = 1, ..., N$ in parallel, we present multiple copies (up to the number of the library size) of it. That is, we generate a collection

$$Q = \{\mathit{\Delta} c(x_1), \mathit{\Delta} c(x_2), ..., \mathit{\Delta} c(x_n), \mathit{\Delta} c(y)\}, \tag{5}$$

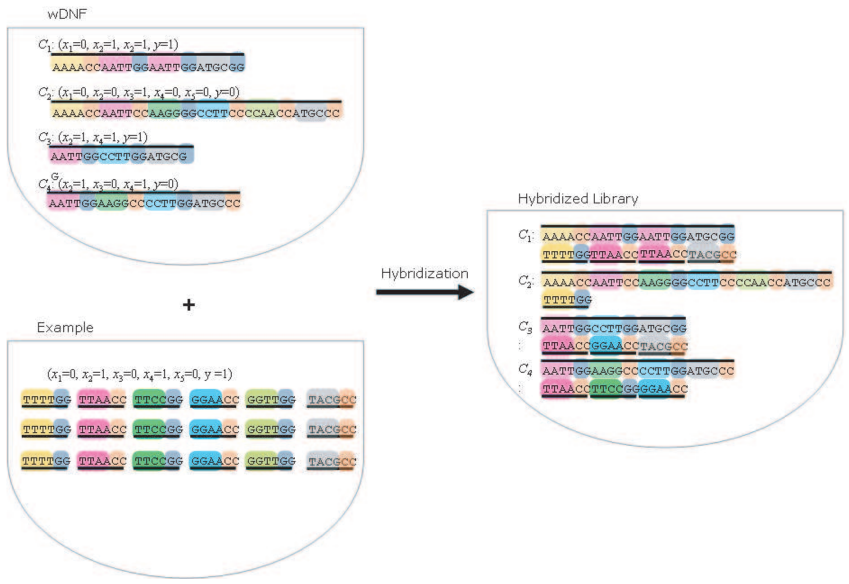
wDNF
C1: (x1=0, x2=1, x2=1, y=1)
AAAACCAATTGGAATTGGATGCGG
C2: (x1=0, x2=0, x3=1, x4=0, x5=0, y=0)
AAAACCAATTCCAAGGGGCCTTCCCCAACCATGCCC
C3: (x2=1, x4=1, y=1)
AATTGGCCTTGGATGCG
C4: (x2=1, x3=0, x4=1, y=0)
AATTGGAAGGCCCCTTGGATGCCC
+
Example
(x1=0, x2=1, x3=0, x4=1, x5=0, y =1)
TTTTGG TTAACC TTCCGG GGAACC GGTTGG TACGCC
Hybridization
Hybridized Library
C1: AAAACCAATTGGAATTGGATGCGG
TTTTGGTTAACCTTAACCTACGCC
C2: AAAACCAATTCCAAGGGGCCTTCCCCAACCATGCCC
TTTTGG
C3 AATTGGCCTTGGATGCGG
TTAACCGGAACCTACGCC
C4 AATTGGAAGGCCCCTTGGATGCCC
TTAACCTTCCGGGGAACC

**Fig. 2.** Illustration of the decision-making procedure using the population of DNA-encoded terms. The query sample is chopped and provided in multiple copies to hybridize in parallel with the terms in the library.

where $\Delta c(\cdot)$ denotes copies made by PCR. The class decision is made by comparing the number of elements in class 1, $N_1$, with that in class 0, $N_0$:

$$y^* = \arg\max_y \{N_y\}, \tag{6}$$

where $y$ takes 0 or 1. The next section discusses a theoretical background for this rule. For learning, we prepare two collections, $M_+$ and $M_-$, consisting of library elements that correctly (or incorrectly) classifies the query sample as follows:

– $M_+ = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of $x_i$ for $i = 1, ..., n$ and $v_i = y\}$
– $M_- = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of $x_i$ for $i = 1, ..., n$ and $v_i \neq y\}$.

Now, we describe how the library is revised to learn from newly observed data. The basic protocol is similar to that described in [13]. The difference is that the DNA molecules are now describing the generalized terms rather than simple examples and the whole test tube represents a wDNF formula. As a new training example $(\mathbf{x}, y)$ is given, we extract from the library the terms whose $x$-part matching with $\mathbf{x}$. The class $y^*$ of $\mathbf{x}$ is determined by the classification procedure described above. Then, the matching terms (library patterns) are modified in their frequency depending on their contribution to the correct or incorrect classification of $\mathbf{x}$. If the label $v$ of the library pattern $(\mathbf{u}, v)$ matching $\mathbf{x}$ is correct, i.e. $v = y$, it is reproduced:

$$L_t \leftarrow L_t + \Delta c(M_+). \tag{7}$$

```
– 1. Let the library $L_0 = \{(\mathbf{u}_i, v_i)\}$ contain the initial wDNF formula. Let $t = 0$.
– 2. Let $t \leftarrow t + 1$.
– 3. Get a training example $(\mathbf{x}, y) = (x_1, x_2, ..., x_n, y)$.
– 4. Let $Q = \{\Delta c(x_1), \Delta c(x_2), ..., \Delta c(x_n), \Delta c(y)\}$.
– 5. Classify $\mathbf{x}$ using $L_t$ as described in the text and construct the following:
    • $M_+ = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of $x_i$ for $i = 1, ..., n$ and $v_i = y\}$.
    • $M_- = \{(\mathbf{u}_i, v_i) | \mathbf{u}_i$ consists of $x_i$ for $i = 1, ..., n$ and $v_i \neq y\}$.
– 6. Update the library $L$ as follows:
    • $L_t \leftarrow L_{t-1} + Q$.
    • $L_t \leftarrow L_t + \Delta c(M_+)$.
      Optionally, $L_t \leftarrow L_t - \Delta c(M_-)$.
– 7. Go to Step 2 if not terminated.
```

**Fig. 3.** The molecular programming (MP) procedure for learning a wDNF formula from examples.

If the label $v$ is incorrect, i.e. $v \neq y$, the matching library pattern is removed from the library:

$$L_t \leftarrow L_t - \Delta c(M_-). \tag{8}$$

Figure 3 summarizes the molecular algorithm for learning a wDNF formula. Note that the library represents a kind of associative memory learned from data. In contrast to other molecular computation models of associated memory [2, 3, 6] proposed so far, the wDNF models contain higher-order patterns. An explicit probabilistic semantics underlying wDNF is also distinguished from other related work.

## 4 The Molecular wDNF Machine as a Probabilistic Computer

We consider the hybridization reaction between two single-stranded DNA molecules $\mathbf{x}_i$ and $\mathbf{x}_q$. Without loss of generality we consider $\mathbf{x}_i$ as the $i$th element (a term in wDNF) in the library and $\mathbf{x}_q$ as a query data. The probability of the $i$th term being retrieved by the query pattern is then expressed as Boltzmann distribution

$$P(\mathbf{x}_i|\mathbf{x}_q) = \frac{\exp\left(-\Delta G(\mathbf{x}_i|\mathbf{x}_q)/k_B T\right)}{\sum_j \exp\left(-\Delta G(\mathbf{x}_j|\mathbf{x}_q)/k_B T\right)}, \tag{9}$$

where $j$ runs over the possible states of hybridization at the absolute temperature $T$ [12]. $\Delta G$ is the Gibbs free energy change for the hybridization reaction and $k_B$ is the Boltzmann constant [9].

A direct computation of this probability is difficult. However, we can approximate this by a Monte Carlo method performed "*in vitro*". To do this, we duplicate the molecules, both $\mathbf{x}_i$ and $\mathbf{x}_q$, let them hybridize, and count the double-stranded DNA at a fixed temperature $T$ below the melting temperature

$T_m$. The estimated value is obtained by averaging the values over the sample of size $|S|$:

$$P(\mathbf{x}_i|\mathbf{x}_q) \approx \frac{1}{|S|}\sum_{i=1}^{|S|}\rho(\mathbf{x}_i, \mathbf{x}_q), \tag{10}$$

where $\rho(\mathbf{x}_i, \mathbf{x}_q) = 1$ if $i$ and $q$ form a double-strand, and $\rho(\mathbf{x}_i, \mathbf{x}_q) = 0$ otherwise. The approximation can be made arbitrarily accurate by increasing the number of copies of the molecules.

By generalizing the above idea of "molecular" Monte Carlo simulation into the collection $L$ of terms, $\mathbf{x}_i$, and a collection $Q$ of a excessive number of the query pattern, $\mathbf{x}_q$, we can compute the probability distribution over the term patterns matching with a query pattern by

$$P_L(X|\mathbf{x}_q) \approx \frac{1}{|L|}\sum_{i=1}^{|L|}P(\mathbf{x}_i|\mathbf{x}_q), \tag{11}$$

where we assume that an excessive number of query molecules are put into the test tube so that all the terms have a fair chance of hybridizing with a query.

We now consider the library representing a $k$-wDNF, the wDNF formula with terms consisting only of $k$ variables of degree 1. The $i$th molecule representing a term with $k$ variables can be considered as a point estimator $f_i^{(k)}(X_1, X_2, ..., X_n, Y)$ of the probability distribution $P_L(X, Y)$. The whole library can then be thought of as a table representing the empirical distribution of the patterns

$$P_L(X, Y) \approx \frac{1}{|L|}\sum_{i=1}^{|L|} f_i^{(k)}(X_1, X_2, ..., X_n, Y), \tag{12}$$

$$f_i^{(k)}(X_1, X_2, ..., X_n, Y) = \frac{\exp\left(-\Delta G(X_1, X_2, ..., X_n|\mathbf{x}_q)/k_B T\right)}{\sum_j \exp\left(-\Delta G(X_1, X_2, ..., X_n|\mathbf{x}_q)/k_B T\right)}, \tag{13}$$

where $\Delta G$ is the Gibbs free energy change for the hybridization reaction and $k_B$ is the Boltzmann constant.

Given the statistical physical interpretation of DNA hybridization and the wDNF representation as an empirical probability distribution, the learning process can be formulated in a probabilistic framework. The objective of learning is to find a wDNF or the library $L$ that best predicts the output label $y$ given input variables $\mathbf{x}$ for all possible training data $(\mathbf{x}, y)$ in the problem space $(X, Y)$. The $L$ can be found iteratively by starting with an initial $L_0$ and updating it as new sample $\mathbf{x}$ is observed:

$$\begin{aligned} L^* &= \arg\max_L P(L|\mathbf{x}, y) = \arg\max_L \frac{P(\mathbf{x}, y|L)P(L)}{P(\mathbf{x}, y)} \\ &= \arg\max_L P(\mathbf{x}, y|L)P(L) = \arg\max_L P_L(\mathbf{x}, y)P(L), \end{aligned} \tag{14}$$

where we used the Bayes rule $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$.

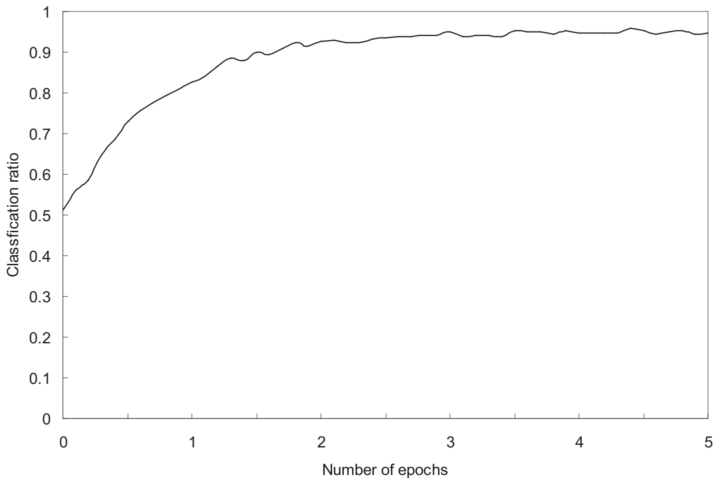

**Fig. 4.** Learning curve of the complete wDNF library. Shown are the average values for 12 runs. Parameters used are the learning rate = $10^{-2}$ and $\beta = 20$.

Once an $L$ is given, the best class label for a query pattern can be determined by computing the probability of each class conditional on the input pattern $\mathbf{x}$, and then determining the class whose conditional probability is the highest, i.e.

$$y^* = \arg \max_{Y \in \{0,1\}} P_L(Y|\mathbf{x}) = \arg \max_{Y \in \{0,1\}} \frac{P_L(Y, \mathbf{x})}{P(\mathbf{x})}, \tag{15}$$

where we used the relation $P(A, B) = P(A|B)P(B)$ and $Y$ represents the candidate classes.

## 5 Simulation Results and Discussion

We evaluated this method on a real-life medical diagnosis problem in simulation. Gene expression data are collected from microarray experiments for AML/ALL leukemia [4]. The microarray data are preprocessed and 10 genes were selected out of 12600 genes. The training set consists of 120 examples each consisting of 10 genes plus the associated leukemia class. A 6-fold cross-validation is used for testing the performance. That is, the whole data set of 120 examples is partitioned into 6 subsets and a total of six learning trials are executed, where each trial used a subset of 20 examples for test and the remaining 100 examples for training. The library was initialized to contain each and every term of wDNF on the 10 variables. These include $(x_1 = 0, y = 0), (x_1 = 0, y = 1), (x_1 = 1, y = 0), (x_1 = 1, y = 1), (x_1 = 0, x_2 = 0, y = 0), (x_1 = 0, x_2 = 0, y = 1), (x_1 = 1, x_2 = 0, y = 0), \ldots$. Thus, the total number of the different library elements is $N = \sum_{k=1}^{10} {}_{10}\mathrm{C}_k \cdot 2^k \cdot 2 = 118,096$, where ${}_{10}\mathrm{C}_k$ denotes the number of cases

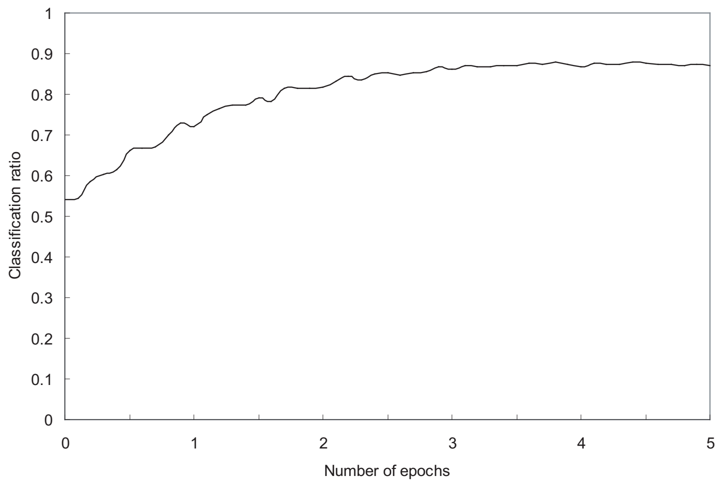

**Fig. 5.** Learning curve of the partial wDNF libraries (average over 12 runs each) containing 10 % of the complete wDNF. Same parameters as in Figure 4

choosing $k$ variables out of 10. For the simulation of in vitro computation of the wDNF formula, we used the library size of $118,096 \times 10^6$, i.e. the initial library was generated by copying each element $10^6$ times. Thus, the library consists of multiple copies of the same terms and we evolved the distributions of the terms through the molecular programming procedure.

For decision making, we used a sigmoid squashing function:

$$f(x) = \frac{1}{1 + \exp(-\beta x)} \tag{16}$$

where $\beta$ is a constant which reflects the level of noise and sets the decision boundary. As mentioned in the previous section, we count the number of each term which answers positive or negative. Then, the proportion of the positives and the negatives is calculated. This result is the input to the sigmoid function. We make a decision probabilistically based on the output of the sigmoid function.

Figure 4 shows the evolution of the performance as learning proceeds. We presented the positive training example and the negative example alternatingly. It should be mentioned that one generation consists of the presentation of one positive and one negative example. The performance was measured at every generation, i.e. each time a pair of new training examples was observed. One sweep through the training set constitutes an *epoch* which is equivalent to 60 generations in this experiment. The best performance of approximately 95% correct classification on the test data set was obtained in 2 epochs.

It is observed that the total number of the different library elements grows exponentially as we allow higher-order terms in the wDNF formulae. That is, if the dimension of the query is high, the total number of the different library

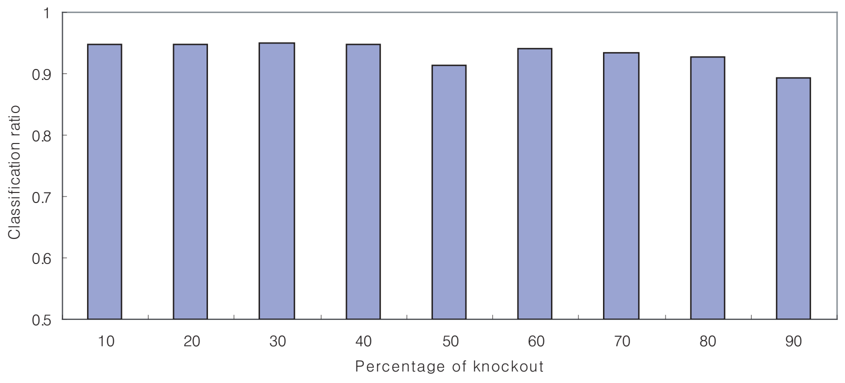

**Fig. 6.** The performance of partial libraries in which some portion of libraries are eliminated after 5 epochs. From left to right 10%, 20%, ..., 90% of whole libraries are eliminated.

elements grows very rapidly. Considering this, it is interesting to know how the total number of library elements affects the performance of wDNF formulae. In order to see how much the performance of wDNF formulae dependends on the complexity of structures, we ran simulations with partial libraries which are generated by eliminating some terms from the complete library. The results are shown in Figure 5. As expected, the wDNF formulae with partial terms perform less than the complete wDNF formulae. However, the results of the partial libraries are still robust. In particular, in the extreme case of the partial library consisting of only 10 % of the full combinatorial terms achieved approximately 90% in absolute accuracy. Figure 6 compares the performances of different partial libraries made by knocking out 10 % to 90 %. The performance was measured in 5 epochs. These results clearly show that the full library is not absolutely necessary to solve this real-life problem using wDNF formulae, suggesting the potential for robust decision making in vitro experiments.

## 6 Conclusion

We introduced the weighted disjunctive normal form (wDNF) as a scheme for representing probability distributions and presented a method for learning a wDNF formula from examples. The learning approach is distinguished from other DNA computing tasks in that the computational result here is a program or machine that can be reused for solving multiple instances of the problem. As the genetic programming provides an automatic programming method for digital computers, the molecular programming provides a method for automatic programming of molecular computers, in our case a wDNF machine.

The results on the leukemia diagnosis problem show that effective solution is possible using the wDNF learning. In particular, our simulation results were competitive to existing state-of-the-art machine learning algorithms. This is somewhat surprising considering the fact that the terms are random conjunctive combinations of Boolean variables. Our analysis suggests that even though the individual terms are simple, their collection as a whole, i.e. wDNF, has a weighted, ensemble representation with redundancy that leads to error-resilient decision making.

Our results on DNA-based diagnosis also suggest a potential use of the molecular learning method for automatically deriving decision rules from wet DNA data. Recently, Benenson et al. [1] demonstrate the possibility of in vitro or in vivo diagnosis. Here the decision rules for diagnosis are hard-coded by the designer. The wDNF learning approach may provide a further step forward into this direction of research by providing a potential means for automatically constructing the robust decision rules from raw data.

## 7 Acknowledgements

This research was supported by the Molecular Evolutionary Computing (MEC) Project of MICE and by the National Research Laboratory (NRL) Program of MOST.

## References

1. Benenson, Y., Gil, B., Ben-Dor, U., Adar, R., Shapiro, E. "An autonomous molecular computer for logical control of gene expression," *Nature*, 429, 423-429, 2004.
2. Baum, E. B., "Building an associative memory vastly larger than the brain," *Science*, 268:583-585, 1995.
3. Chen, J. Deaton, R. and Wang, Y.-Z., "A DNA-based memory with in vitro learning and associative recall," DNA9, *Lecture Notes in Computer Science* 2943:145-156, 2004.
4. Cheok, M.; et al., "Treatment-specific changes in gene expression discriminate in vivo drug response in human leukemia cells," *Nature Genetics*, 34:85-90, 2003.
5. Koza, J. R., *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, MA, USA, 1992.
6. Reif, J.H., LaBean, T.H., Pirrung, M., Rana, V.S., Guo, B., Kingsford, C., Wickham, G.S., "Experimental construction of very large scale DNA databases with associative search capability," DNA7, *Lecture Notes in Computer Science* 2340:231-247, 2002.
7. Rose, J. A., Deaton, R. J., Hagiya, M., Suyama, A., "A DNA-based in vitro genetic program", *Journal of Biological Physics*, 28:493-498, 2002.
8. Sakakibara, Y., "Solving computational learning problems with Boolean formulae on DNA computers," DNA6, *Lecture Notes in Computer Science*, 2052:220-230, 2001.
9. SantaLucia, J. and Hicks, D., "The thermodynamics of DNA structural motifs," *Annu. Rev. Biophys. Biomol. Struct.*, 33:415-440, 2004.
10. Shackelford, G. and Volper, D., "Learning k-DNF with noise in the attributes," *COLT '88: Proc. First Annual Workshop on Computational Learning Theory*, 97-103, 1988.

11. Valiant, L., "Robust logics", *Proc. ACM Symposium on the Theory of Computing* (STOC 99), pp. 642-651, 1999.
12. Wartel, R.M. and Benight, A.S., "Thermal denaturation of DNA molecules: A comparison of theory with experiments," *Physics Reports*, 126(2):67-107, 1985.
13. Zhang, B.-T. and Jang, H.-Y., "A Bayesian algorithm for in vitro molecular evolution of pattern classifiers," *Proc. of 10th Int. Meeting on DNA Computing*, Milan, Italy, pp. 294-303, 2004.
14. Zhang, B.-T. and Müehlenbein, H., "Balancing accuracy and parsimony in genetic programming," *Evolutionary Computation*, 3(1):17-38, 1995.